Suboptimal Minimum Cluster Volume Cover-Based Method for Measuring Fractal Dimension

Charles R. Tolle, *Senior Member*, *IEEE*, Timothy R. McJunkin, *Member*, *IEEE*, and David J. Gorsich

Abstract—A new method for calculating fractal dimension is developed in this paper. The method is based on the box dimension concept; however, it involves direct estimation of a suboptimal covering of the data set of interest. By finding a suboptimal cover, this method is better able to estimate the required number of covering elements for a given cover size than is the standard box counting algorithm. Moreover, any decrease in the error of the covering element count directly increases the accuracy of the fractal dimension estimation. In general, our method represents a mathematical dual to the standard box counting algorithm by not solving for the number of boxes used to cover a data set given the size of the box. Instead, the method chooses the number of covering elements and then proceeds to find the placement of smallest hyperellipsoids that fully covers the data set. This method involves a variant of the Fuzzy-C Means clustering algorithm, as well as the use of the Minimum Cluster Volume clustering algorithm. A variety of fractal dimension estimators using this suboptimal covering method are discussed. Finally, these methods are compared to the standard box counting algorithm and wavelet-decomposition methods for calculating fractal dimension by using one-dimensional cantor dust sets and a set of standard Brownian random fractal images.

Index Terms—Fractal dimension, Fuzzy-C means, suboptimal cover, box counting, clustering, texture analysis.

1 Introduction

This paper extends prior work [1], [2], [3] on developing more efficient and accurate methods for determining fractal dimension. Our interest in fractal dimension is primarily due to its ability to segment images into different textural regions. Textural analysis of subregions is important because it is connected to current models of the human visual system (HVS). Texture perception and, more specifically, texture roughness is a key cue feature used in recognition of objects [4], [5]. In [2], we began investigating the use of fractal dimension in segmenting images. Jardine [4] found that the HVS perceives changes in roughness of textures corresponding to the fractal dimension changes of random fractal textures. Likewise, at Lincoln Laboratory, the fractal dimension was shown to be one of the best five cue features (out of 13 studied) for use in the automatic target recognition algorithms of SAR imagery [5]. Moreover, the calculation of fractal dimension has a natural extension to the general sensor fusion problem. For example, once a series of spectral images/signals for an area have been registered (overlaid) within a *fused* high-dimensional space the process of calculating the fractal dimension for that space is clearly understood and easily accomplished. The computational procedure discussed in this paper is also used in obtaining an additional important textural feature known as lacunarity [6], [7]. Lacunarity describes the spatial gaps within a data set, i.e., how the space was filled, while fractal dimension describes the space-filling nature/capabilities of a data set, i.e., the amount of space filled [6], [7].

A fractal is a set that has a noninteger fractal dimension. The fractal dimension is formally defined as the Hausdorff-Besicovitch (HB) dimension. However, there are a number of ways to estimate this fractal dimension, each of which uses a slightly different definition of the dimension. A few of these methods are: box algorithm, wavelet transform, power spectrum method (using the Fourier transform), Hurst coefficients, Bouligand-Minkowski [8], the variation method [9], and capacity dimension. Recent work has investigated which of these methods are efficient versus accurate [10]. In addition, many works have pursued improving the accuracy of these techniques [12], [13]. For the most part, these efforts have been limited to improvements in efficiency and accuracy for time varying signals as well as geometrically produced images. We expand upon this body of work by adding our suboptimal fractal dimension estimator, which is extensible to any arbitrary L dimension space. Within this paper, however, we focus on texture images with a known fractal dimension. (For a quick and easy introduction to calculating fractal dimension of images, we suggest that the reader start with an online program Web hosted by Charles Sturt University: http://life.csu.edu.au/fractop/.)

Although most commonly used fractal dimension approximation methods run very efficiently [2], [6], [10], [14], the actual estimates for these algorithms suffer from differing amounts of inaccuracies. Due to these inaccuracies, we decided to develop a new way of estimating the fractal dimension that is much closer to the definition of the box dimension than is the commonly implemented via the box counting algorithm. Our hypothesis was that by obtaining a

- *C.R. Tolle and T.R. McJunkin are with the Idaho National Engineering and Environmental Laboratory, PO Box 1625, Idaho Falls, ID 83415-2210. E-mail: {tollcr, mcjutr}@inel.gov.*
- *D.J. Gorsich is with the US Army Tank-Automotive, Research, Development & Engineering Center Robotics Laboratory, MS 263, Warren, MI 48397-5000. E-mail: gorsichd@tacom.army.mil.*

Manuscript received 14 May 2001; revised 13 Feb. 2002; accepted 28 June 2002. Recommended for acceptance by A. Khotanzad.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 114147.

better estimate of the optimal cover required for determining the box dimension, the fractal dimension estimates would become more accurate. To achieve this end, we introduce a dual mathematical concept of solving the optimal cover problem normally stated when defining the fractal dimension. Instead of choosing a fixed covering element size and then estimating the positions and number of covering elements that optimally cover the set of interest, we reverse the process by first choosing the number of covering elements and then solve for the covering element size. This initial concept was presented with two possible solutions in [1]. The first solution solved the covering problem via a Fuzzy-C Means (FCM) clustering algorithm, while the second solution utilized a genetic algorithm. To test the accuracy of these new methods, a texture generation program given by Musgrave in Ebert's book [15] was used. The texture generator was designed to generate random brownian fractal textures given a fractal dimension and a lacunarity (see Fig. 2). The same type of image will be used in this paper to check the accuracy of our new fractal dimension estimator.

As in [1], we explore a method for using clustering algorithms to determine a suboptimal cover which is then used to estimate the fractal dimension of a data set. In this paper, we introduce the use of a Minimum Cluster Volume (MCV) clustering algorithm [18] for finding the suboptimal cover. The MCV cost surface is more complex due to its numerous local minima [18]. To overcome this limitation, we have chosen to initialize the MCV center locations using a modified FCM algorithm, instead of the more traditional method of randomly selecting initial center positions (for completeness the modified FCM algorithm is discussed in Appendix A). A further evolutionary step beyond our initial paper [1] is the use of singular values of the cluster scatter matrices to better describe the size and shape of the covering elements. Moreover, averages of these singular values are used to estimate a more traditional covering element size used within the calculation of the box dimension. A number of element size measurements are considered, e.g., the minimum of the maximum of the singular values for each cluster (MIN-MAX) and the maximum of the maximum of the singular values for each cluster (MAX-MAX).

In the section that follows, the Hausdorff-Besicovitch (HB) dimension is defined, leading to a simpler definition of the box dimension. From the detailed discussion of the box dimension and its well-known algorithm, we develop a dual mathematical method for obtaining an optimal cover that will be used to estimate the box dimension with greater accuracy. At that point in the paper, we introduce the new MCV suboptimal fractal dimension estimators. Next, detailed sets of results are presented for a series of cantor dust sets and Brownian random fractal images. Finally, we close with a description of the future directions of this research.

2 Fractal Dimensions

The fractal dimension is most commonly defined as the Hausdorff-Besicovitch (HB) dimension, $D_h(A)$, where A denotes the image/signal. In general, the HB dimension of A is defined in the following manner [16].

Let

$$R^n = \{\mathbf{x}|\mathbf{x} = (x_1, \ldots, x_n), x_i \in R\}, \tag{1}$$

for some natural number n. Then, define the diameter of some cover C as

$$diam(C) = sup\{d_e(x, y)|x, y \in C\}, \tag{2}$$

where $d_e(x, y)$ denotes the Euclidean distance function. An open cover of A is defined by covers C_i such that

$$A \subset \bigcup_{i=1}^{\infty} C_i. \tag{3}$$

Then, let

$$h_\epsilon^s(A) = \inf\left\{\sum_{i=0}^{\infty} diam(C_i)^s \middle| \overset{\{C_1,C_2,\} \text{ open cover}}{of\ A\ with\ diam(C_i) \leq \epsilon}\right\}, \tag{4}$$

so that the s-dimensional Hausdroff measure of A is

$$h^s(A) = \lim_{\epsilon \to 0} h_\epsilon^s(A). \tag{5}$$

Then, the Hausdorff-Besicovitch (HB) fractal dimension of A is defined as

$$D_h(A) = \inf\{s|h^s(A) = 0\} = \sup\{s|h^s(A) = \infty\}. \tag{6}$$

Due to the complexity and impracticality of finding the optimal cover defined by the HB dimension, we need a different bounding estimate, i.e., one that is easily calculated. One such upper bound estimation of the HB dimension is the box dimension definition.

2.1 Using the Box Counting Algorithm and Other Methods to Determine Fractal Dimension

The box dimension, $D_b(A)$, which is normally estimated using the box counting algorithm, is a commonly used upper-bound estimator for the HB dimension. In general, the box dimension can be defined as follows [16].

Let $\mathcal{C}_d(A)$ be the smallest number of closed covering elements of size d, see (2), that cover the set A. Then, $D_b(A)$ is defined as

$$D_b(A) = \lim_{d \to 0} \frac{\log \mathcal{C}_d(A)}{\log \frac{1}{d}}. \tag{7}$$

It is important to note that the box dimension, $D_b(A)$, does not always equal the HB dimension, $D_h(A)$. For example, it can be shown that $D_b(A) = n$ for any dense subset A such that $A \subset R^n = \{x|x = (x_1, \ldots, x_n), x_i \in R\}$. Likewise, for the same A, $D_h(A) \leq n$. Moreover, $D_h(A) = 0$ for any such countable set A [16]. Therefore, given the set A of rational numbers on $[0, 1]$, the box dimension is $D_b(A) = 1$, while the HB dimension is $D_h(A) = 0$. Although the box dimension fails in some instances, the value it normally produces is similar to the HB dimension.

The calculation of the box dimension is often difficult. One must first find the optimal covering (smallest number of box shaped covering elements) of A for a given set of box covering elements with edges of length d. Placing the covering elements in an algorithmic way to minimize their total number for a given size is easily achieved for low-dimensional data sets. However, this process becomes much less tractable in the case of more general forms of data. This also requires algorithms to determine when and how to overlap covering elements when a whole number of elements does not "evenly" fit the space. Fig. 1 shows a simple example that

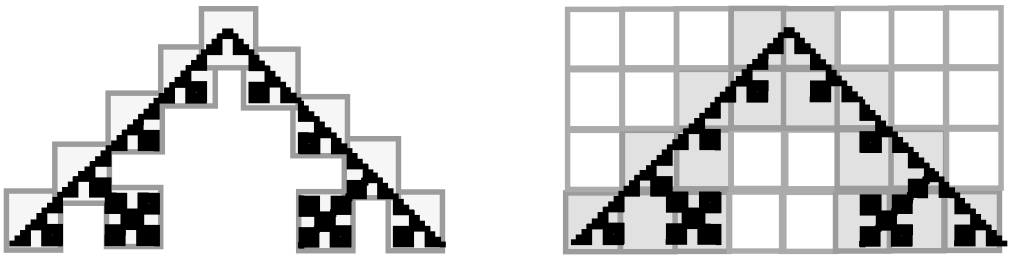

Fig. 1. An example of two possible coverings of a geometric fractal using simple box covering elements; on the left is an optimal placement of the elements requiring only 11 elements, and on the right is a typical box counting algorithm placement of the elements requiring 16 elements (a 45 percent increase in the number of elements needed).

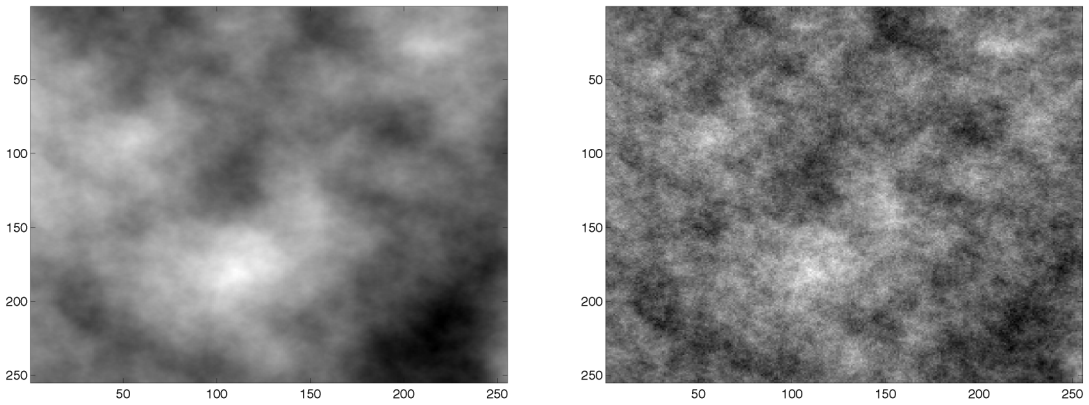

Fig. 2. An example of two Brownian motion fractal textures; the fractal dimension is 2.1 on the left and 2.7 on the right.

starts to express the difficulty of generating the box dimension and its estimate via the box counting algorithm. Note, even within this simple example, a grid-based placement method such as the box algorithm can cause an increase of 45 percent in the number of covering elements needed to cover the set optimally. Even with these inaccuracies, the box dimension is normally estimated using the box counting algorithm because of its simplicity of implementation. In short, the box counting algorithm places a standard grid of boxes (or hypercubes) upon the image/signal and counts the number of nonempty boxes as covering elements. These boxes act as the required open cover, C_i. This count of covering elements is then plotted on a log-log plot versus the reciprocal of element size d in Fig. 3. Finally, the box dimension estimate is taken from the monotonically rising nonzero linear slope of the log-log plot. By examining this procedure closely, one can see that the only difference between the results obtained using the box counting algorithm and the box dimension is due to the fixed location of the potential covering elements as well as their inability to allow for possible overlaps required to achieve an optimal covering, see Fig. 1. In other words, the box counting algorithm does not, in general, find the optimal cover. It has been shown that the box counting algorithm needs at least $10^{D_h(A)}$ points to determine the fractal dimension [17]. The key concept underlying the box counting algorithm is: first, the box size was chosen and then the number of boxes needed to cover the set must be calculated in order to estimate the dimension.

Another common fractal dimension estimator is the Bouligand-Minkowski [8] estimator. A concise overview of this method is contained in [12]. It is traditionally defined for use on R^2 data sets. This estimator defines a cover element size as a radius, ϵ. The complete cover is the union of all covering elements created by centering each element on every point contained within the data set. The variation of the area contained within the cover, $A(\epsilon)$, as ϵ is decreased then provides the measure of the fractal dimension: $D = \lim_{\epsilon \to 0}\left[2 - \frac{\ln A(\epsilon)}{\ln \epsilon}\right]$. Furthermore, as with the box algorithm, the empirical estimates are found by calculating the slope of the cover size $A(\epsilon)$ with respect to the changing element size ϵ. However, as the dimensionality of the data

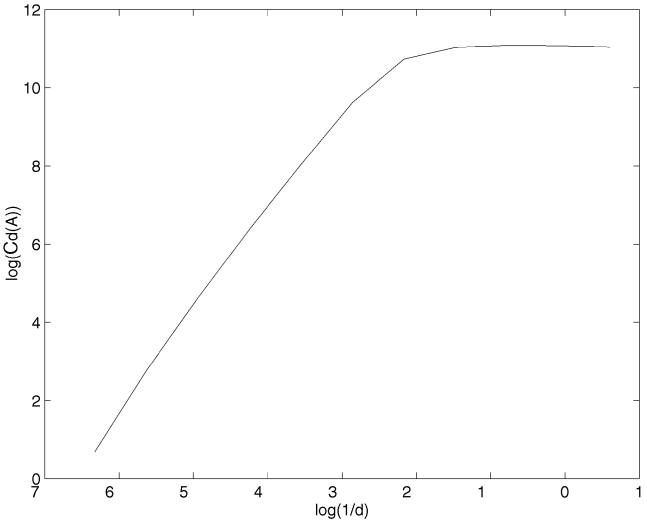

Fig. 3. The log-log plot of the results from the box counting algorithm for the texture with a fractal dimension of 2.7 shown in Fig. 2. The linear slope on the log-log plot indicates a fractal dimension of 2.5.

set grows estimating the hypervolume becomes exceedingly difficult to achieve.

Next, we consider the variation method [8]. This method, like the one developed below, provides a means to estimating fractal dimension of an image, e.g., Fig. 2, where the image intensity or amplitude can be written as a function of a spatial coordinate, $z = f(x, y)$. As before, this method requires the choice of various size ϵ. A covering element of radius ϵ is placed at each data point within the image. Next, the variation at each point, $v_f(x, y, \epsilon)$, within the element is then measured by taking the difference between the maximum, z_{max}, and minimum, z_{min}, functional values contained within the covering elements region. These variations are then averaged over the image to form the ϵ-variation, $V_f(\epsilon)$, for the image. Note, in the finite case, this is a simple summing operation and, in the analytic or general case, this becomes an integral. The fractal dimension definition then takes the form: $D = \lim_{\epsilon \to 0}\left(3 - \frac{\ln V_f(\epsilon)}{\ln \epsilon}\right)$. In order to estimate a finite sampled set, one takes the slope of the plot of $\ln V_f(\epsilon)/\epsilon^3$ versus $\ln 1/\epsilon$. This method has many conceptual overlaps with the method discussed below; however, it is limited to functional hypersurfaces above a coordinate hyperplane. Accuracy of this method for textural fractals obtained by Summers et al. [11] is good about 2.5, however, it varies significantly as the allowable extremes for the method are approached, i.e., 2 and 3.

We now introduce what we will call a dual mathematical form for the box algorithm: instead of finding the minimum number of covering elements for a data set using a given element size, we propose that one can choose the number of covering elements then find the minimum size of each element required to cover the data set. This method limits the problem Zeng et al. [12] calls "Choice of sequence," because the choice of number of covers is strictly contained to the set of natural numbers.

2.2 MCV Suboptimal Cover-Based Fractal Dimension

We start our discussion of the MCV suboptimal cover estimation of the box dimension by exploring the dual mathematical solution. As was mentioned, instead of choosing the covering element size and then finding the number of covering elements for the image, one might just as well choose the number of covering elements and then find the smallest covering element size. Simple clustering methods solve this problem.

Since we are interested in obtaining the smallest clusters that will fully cover the data set, using a minimum cluster volume optimization procedure seems logical. That procedure has been resolved in the MCV clustering problem [18]. The algorithm is briefly described next and described in slightly more detail in Appendix B.

The center positions are initialized using a modified FCM algorithm [19], see Appendix A. The need for seeding the center location for the MCV algorithm is also discussed in Krishnapuram and Kim's paper [18]. The MCV algorithm minimizes the sum of the fuzzy covariance matrix determinants, $|C_{fi}|$, with the same constraint on the membership values, u_{ik}, as the FCM algorithm. This minimizes the sum of products of the matrices' singular values, which is proportional to the hypervolume of the clusters

$$J_{fv} = \min_{u_{ij}, v_i} \sum_{i=1}^{\mathcal{C}} |C_{fi}|^{1/2}, \tag{8}$$

where

$$C_{fi} = \frac{\sum_{j=1}^{N} u_{ij}^m (\mathbf{x}_j - \mathbf{v}_i)(\mathbf{x}_j - \mathbf{v}_i)^T}{\sum_{j=1}^{N} u_{ij}^m}, \tag{9}$$

with the constraints of $\sum_{i=1}^{\mathcal{C}} u_{ji} = 1$ for each point $\mathbf{x}_j$ for $j = 1, \ldots, N$; $u_{ji} \in [0, 1]$; $\mathcal{C}$ is the number of clusters; N is the number of points in the data set; $\mathbf{v}_i$ is the center for cluster, C_i, $i = 1, \ldots, \mathcal{C}$; and $m \in [1, \infty)$ is a fuzzifying exponent. As m increases, the clustering approaches the hard cluster limit. For this paper, we chose m to be 2. One of the strengths of MCV is that it more closely associates the center of the cluster with points within its core, rather than allowing many points with weak membership to effectively "pull" the cluster center away from its *natural* core members, i.e., the points with stronger membership values. This makes MCV robust to outliers within the data set.

To obtain the cover found by solving the MCV algorithm, each data point is assigned to the cluster in which its membership value is highest, that is "hard" clustered. The "hard" clusters define the covering elements. The next step is to determine how best to find the size of each one of these covering elements. The reader should note that, in general, these covering elements are not hyperballs as the HB and box dimension require. Instead, each cluster represents an L-dimensional hyperellipsoid. How one best determines the method for consolidating the L components of each cluster's hyperellipsoid size, as well as the total cover's estimated "ball size," is an open problem. One such method, described in [1], is to choose the average of the maximum distance from each center to any point in that "hard" cluster. This method fits an L-dimensional ball shaped covering element around each center, even if the shape of the hyperellipsoid is not spherical. This does not seem to be the best possible choice. Instead, we suggest another method based on the singular values (SV) of each cluster's scatter matrix.

The average scatter matrix for cluster i is defined as

$$\hat{\mathcal{M}}^i = \frac{1}{N_i} \sum_{\mathbf{x}_k \in C_i} (\mathbf{x}_k - \mathbf{v}_i)(\mathbf{x}_k - \mathbf{v}_i)^T, \tag{10}$$

where N_i are the number of data elements covered by C_i. This matrix can be decomposed into its principal components using the singular value decomposition

$$\hat{\mathcal{M}}^I = W^i \mathcal{Q}^i {G^i}^H \tag{11}$$

$$= W^I \begin{bmatrix} \sigma_1^i & 0 & \cdots & 0 \\ 0 & \sigma_2^i & & 0 \\ & & \vdots & \\ 0 & 0 & \cdots & \sigma_L^i \end{bmatrix} {G^i}^H \tag{12}$$

$$= \sum_{l=1}^{L} \sigma_l^i w_l^i {g_l^i}^H, \tag{13}$$

where ${G^i}^H$ denotes the Hermitian of G^i, σ_l^i are the singular values, and W^i and G^i are unitary matrices, i.e., ${W^i}^H W^i = I$ and $G^i {G^i}^H = I$ (which describe the singular value directions).

These singular values provide the size and shape of the hyperellipsoid covering element located on each cluster

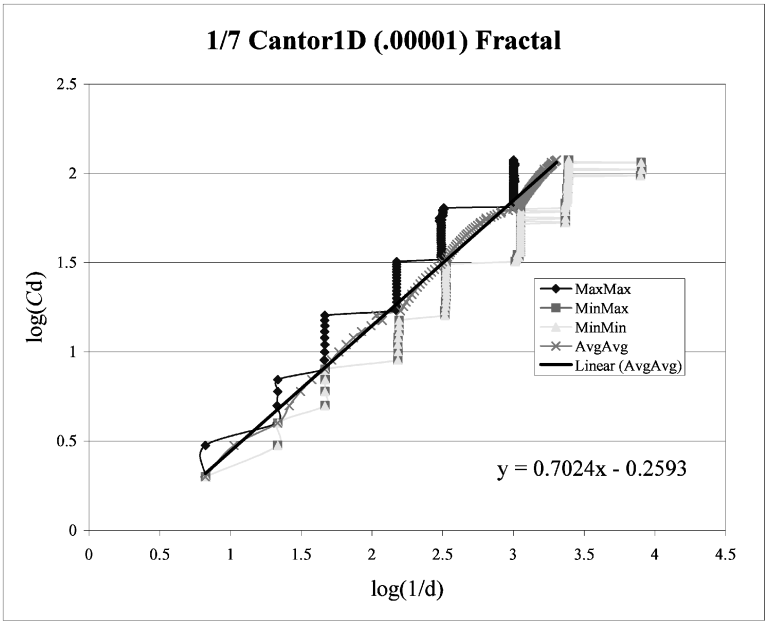

Fig. 4. Typical $log(\mathcal{C})log(\frac{1}{d})$ plots for the Max-Max, Min-Max, Min-Min, and Ave-Ave ball estimations are shown using a $\frac{1}{7}\text{th}$ cantor set.

center. Each covering element can be measured in L different component directions. A particular dimension of the data vector $\mathbf{x}$ could be distributed such that one or more SVs are significantly different in their magnitude from the remaining dimensions. For the data sets investigated here, we present three dimension estimators: the Max-Max method, which estimates the ball size by maximizing over all clusters, i, of the maximum directional SV, σ_l^i, for each cluster; the Min-Max method; and Ave-Ave (AVE=average) method. We highlight the Ave-Ave estimate because it has a more linear $\log(\mathcal{C}_d)/\log(\frac{1}{d})$ trend for highly ramified data sets such as a cantor dust (see Fig. 4). The lack of such a linear trend in the other estimators appears to be due to poor clustering, produced when the number of *true* clusters does not fit the data being clustered. Fig. 5 shows that in nongeometrically self-similar fractal data sets, the Ave-Ave estimate is not merely the only near linear function but the only one which approaches the expected fractal dimension.

Figs. 4 and 5 present a typical $\log(\mathcal{C})$-$\log(\frac{1}{d})$ graph, where d is the *size* of the cluster as determined by the one of the estimators discussed above, e.g., Ave-Ave. For the Ave-Ave, d is calculated as

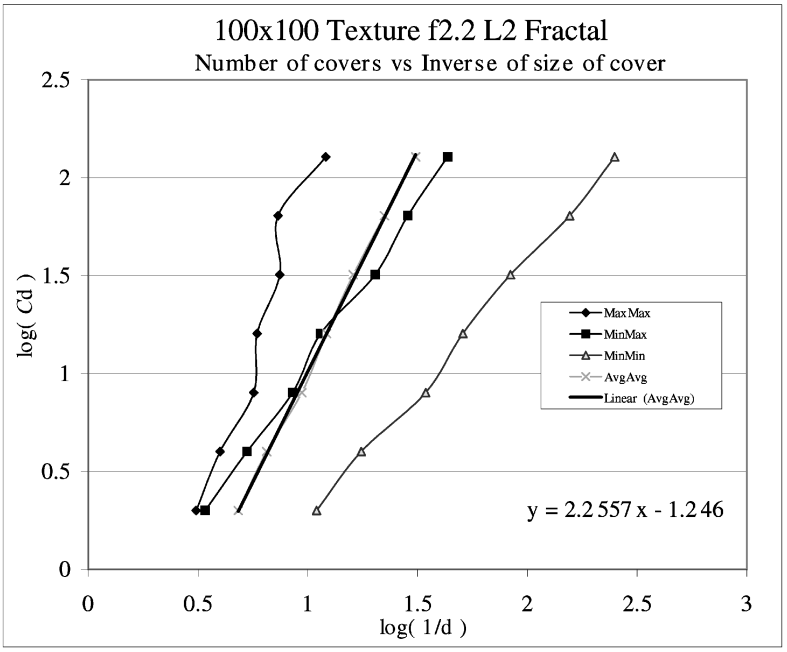

Fig. 5. Typical $log(\mathcal{C})$-$log(\frac{1}{d})$ plots for the Max-Max, Min-Max, Min-Min, and Ave-Ave ball estimations are shown using a Brownian random texture with fractal dimension 2.2 and a lacunarity of 2.

$$d_{Ave\text{-}Ave} = \sqrt{\frac{1}{\mathcal{C}L}\sum_{l=1}^{\mathcal{C}}\sum_{i=1}^{L}\sigma_l^i}, \tag{14}$$

where the square-root is necessary to retrieve a linear distance measure from the singular values. Notice that, for the cantor set, the Max-Max and Min-Min estimations appear to climb in a step fashion, see Fig. 4. In fact, these steps occur with even powers of 5, which corresponds to the number of division of a $1/7$ cantor set at each successive resolution iteration (this graph visually demonstrates the poor clustering problem). Since no a priori knowledge about the structure of a data set is assumed, the average appears to be the best choice. Furthermore, notice that this qualitatively shows the effects of poor clustering due to the natural internal structure of the cantor dust for different resolution levels.

Although this method does not maintain a constant size and shape for each covering element, one can contrive a cover based on the Ave-Ave result that does accomplish this. In order to do this, one must choose a σ such that: one, when a ball shaped covering element of radius $\sigma d_{Ave\text{-}Ave}$ is placed on the calculated cluster center it completely encloses all of the associated cluster members within that cluster; two, this σ must accomplish this for all clusters within a particular covering; three, the σ must further hold over all of the coverings used within the fractal dimension estimation calculation. It should be clear that by choosing such a σ, it also meets the requirement needed for all $\mathcal{C}_d$, therefore, since the slope $\log\frac{1}{\sigma d} = -\log\sigma - \log d$ and the slope of $\log(\mathcal{C})$-$\log(\frac{1}{\sigma d})$ does not change with σ, the fractal dimension estimate will not change, thus this procedure is unnecessary. Moreover, by using the Ave-Ave statistic for measuring the variation of the SVs of the clustering elements, one obtains an implicit smoothing function that overcomes misfit clusters that could adversely affect the fractal dimension measurement. We show in the next section, that the measurement of this statistic of the covering element sizes (although not strictly in the spirit of Hausdorff-Besicovitch) provides a tractable, reliable method for approximating fractal dimension on many types of data sets. We concede that a method for clustering while maintaining strict size conformity of the clusters would give a better upper bound for the box dimension, however, to date the authors have not found a method to do this in general. Even with this concession, this method finds a strong ally in the variation method discussed earlier, in that both methods conceptually measure the variation within the covering elements to estimate the fractal dimension.

3 Results

In the previous section, we introduced a new near-optimal cover for estimating the box dimension. This new MCV suboptimal cover will be shown to improve the estimation of the box dimension over the traditional box counting algorithm. However, what we gain in accuracy, one might think we give up in computation requirements. The results shown next quite clearly demonstrate that our new method makes a much better upper-bound estimate of the theoretical fractal dimension. Moreover, these computations are the basis for another method of calculating lacunarity [7], so they might not be so unreasonable when one is using both fractal dimension and lacunarity to more fully describe the textural qualities of a data set. With this in mind, the computation-conscious reader should note that these algorithms were run on a dual Intel Pentium III 450 Mhz platform running Redhat 7.0 Linux using the CILK-5 [22], [23] compiler out of Massachusetts Institute of Technology to provide parallel thread management. On this platform, a typical 10,000 point image file took about 24 hours to complete. So, the method discussed here is orders of magnitude more computationally complex than the traditional box counting algorithm. The same image file on the same platform, our own highly computationally efficient box counting algorithm, also written in CILK, took less then a minute. Even with this stark difference in computation time, the accuracy achieved using this new algorithm is well worth the computational costs given the appropriate offline application (remember one can also obtain lacunarity at the same time [7]). If speed is an issue, one can use the FCM clustering algorithm to obtain the desired cover and still achieve a high level of accuracy.

To show the accuracy of these new methods, we use a standard set of Brownian random fractal texture images produced using Musgrave's texture generation program [15]. These images have theoretical fractal dimensions ranging from 2.1 to 2.9 (an example is shown in Fig. 2). This texture generation program also controls the theoretical lacunarity of the images. As mentioned earlier, by changing the lacunarity of a texture, one can further vary its appearance while maintaining the fractal dimension. We provide fractal dimension estimations for a number of lacunarities ranging from 2 to 10 in the texture generation algorithm. In this paper, we refer to the word *virtual* as a method of creating and evaluating three-dimensional Brownian random textures generated by the texture generation algorithm in their nonscaled floating point form. With the use of these *virtual* images, instead of gray-scale images such as shown in Fig. 2, it is hoped that any effects that might change the theoretical fractal dimension in the scaling and discretizing process used to create the gray-scale image will be minimized. In addition to the three-dimensional texture surfaces, one-dimensional cantor sets are also used to test the algorithms. These were chosen because the cantor set has a well-understood and easily calculated theoretical fractal similarity dimension, i.e., the cantor set's fractal dimension is defined as the logarithm of the number of segments kept at each iteration, divided by the logarithm of the number of subdivisions made at each iteration as the set is created. For example, the $\frac{1}{3}$ cantor set has a fractal dimension of $\frac{\log(2)}{\log(3)}$. Tables 1, 2, and 3 show the results for each of the methods that have been discussed thus far.

Notice that the results of the new algorithm fall within a six percent band of the theoretical fractal dimension for all the textures processed. In comparison, the box algorithm's accuracy decreases greatly as the fractal dimension increases. In fact, even to get in the ballpark of the theoretical fractal dimension as it approaches 3, the box counting algorithm requires an additional scaling factor not included within the theoretical derivation of the algorithm. The box counting algorithm appears to vary greatly with changes in lacunarity. (Note: the wavelet algorithm also requires scaling to provide numerically accurate results.) It is true that a difference of fractal dimension on the order of 0.1 is difficult to distinguish either visually or via algorithm; however, this should be

TABLE 1
Estimated Fractal Dimensions for Simple Cantor Dust Sets

Number of Points	Cantor: Removals / Divisions	Theoretical Fractal Dimension	Box Algorithm Estimate	MCV using Average of singular values
781	1/3	0.631	0.622	0.631
952	2/5	0.683	0.683	0.683
1971	3/7	0.712	0.706	0.702

TABLE 2
Estimated Fractal Dimensions for the Virtual Brownian Fractal Textures with Common Lacunarity

Fractal Texture Sampling	Lacunarity	Theoretical Fractal Dimension	Box Algorithm Estimate	Wavelet Based Estimate	MCV using Ave-Ave singular values
100x100	4	2.1	2.092	2.497	2.078
100x100	4	2.2	2.118	2.507	2.256
100x100	4	2.4	2.186	2.534	2.290
100x100	4	2.7	2.324	2.650	2.705
100x100	4	2.9	2.377	2.762	3.003

expected given their closeness in roughness. Note: we acknowledge that results of greater than 3.0 on a set with topological dimension of 3 is impossible; however, the result is within the typical percentage error and is thus accepted as a reasonable result of the method.

For completeness, results of the fractal dimension calculation on clusters found using only the FCM algorithm are included for some of the data. The FCM accuracy is somewhat degraded compared to MCV. In some instances, the results fall outside the six percent bin; however, because the FCM algorithm does not require inverting covariance matrices, FCM takes less than one hour to process for the same data sets.

In the case of the much simpler cantor dust sets, both algorithms appear to closely reproduce the theoretical fractal dimensions. Based only on the heuristic results within this study, one might expect that as the number of dimensions within the data set increases, the accuracy gap will also widen. Of course, we only present the results for simple Brownian random fractal textures; however, within the real world of target detection one is interested in much higher dimensional fused multispectral images where this widening gap might become a large stumbling block for the use of traditional fractal dimension estimators as a quality cue feature. This is something that the reader should consider when choosing a method of calculating fractal dimension within high-dimensional data sets.

4 Conclusions and Future Directions

By re-examining the box dimension definition, a mathematical dual problem for its estimation based on the box counting algorithm was presented. This dual problem assumes the number of covering elements (hyperballs) is known and one only need to find the optimal placement and size of the covering elements. This problem can be restated as the MCV clustering problem. We have chosen to modify the FCM clustering algorithm in order to obtain a better set of starting conditions for use with the MCV algorithm. The MCV algorithm is then used to obtain the near-optimal cover needed to estimate the box dimension. Several methods for estimating the covering elements' ball sizes have been developed for clustering-based algorithms such as FCM and MCV, with the Ave-Ave method appearing to be the most robust for general types of data. Most importantly, the estimates made using the new MCV suboptimal fractal dimension estimation algorithm presented here are significantly more accurate than those of the traditional box counting algorithm and the wavelet algorithm. This method also provides a tractable solution for data sets, which do not have a analytical solution using traditional methods like

TABLE 3
Estimated Fractal Dimensions for the Virtual Brownian Textures with same Fractal Dimension and Varying Lacunarity

Fractal Texture Sampling	Lacunarity	Theoretical Fractal Dimension	Box Algorithm Estimate using Ave-Ave	MCV using Ave-Ave singular values	FCM only Ave-Ave singular values
100x100	2	2.2	2.098	2.256	2.191
100x100	4	2.2	2.118	2.097	2.096
100x100	7	2.2	2.128	2.182	2.051
100x100	10	2.2	2.084	2.117	2.045
100x100	2	2.9	2.247	2.914	2.858
100x100	4	2.9	2.377	3.003	2.984
100x100	7	2.9	2.407	3.036	3.034
100x100	10	2.9	2.264	2.837	2.774

Hausdorff-Besicovitch or Bouligand-Minkowski. Also, the numerical accuracy of this new algorithm gives us confidence to proceed into higher dimensional data sets, where an image such as the Brownian textures is not a readily available for scaling calibration. Although work continues to significantly improve the computational speed of this new estimation algorithm, it shares its computational load almost completely with another important textural cue feature lacunarity [7]. This fact helps "pay" for the computational tradeoff and should be considered for the appropriate offline classification projects or real-time classification given powerful dedicated hardware.

APPENDIX A

FINDING AN INITIAL SUBOPTIMAL COVER: A MODIFIED FUZZY-C MEANS PICARD ITERATION

The Fuzzy-C Means (FCM) algorithm solves the following problem:

$$J_{FCM} = \min_{u_{ik},\mathbf{v}_i} \sum_{i=1}^{\mathcal{C}} \sum_{k=1}^{N} u_{ik}^2 \|\mathbf{x}_k - \mathbf{v}_i\|^2, \tag{15}$$

with constraints: $u_{ik} \in [0,1]$, $\sum_{i=1}^{\mathcal{C}} u_{ik} = 1 \; \forall k = 1, 2, \ldots, N$; and where $\mathcal{C}$ is the number of clusters, N is the number data vectors $\{\mathbf{x}_k\}$ being clustered, u_{ik} is the membership value of the kth data vector in the ith cluster, and $\mathbf{v}_i$ is center (mean) vector of the ith cluster. The $\| \bullet \|$ is the Euclidean norm operator. It has been shown [19] that this problem can be solved using the Picard iteration [20], also known as the method of successive approximations [21], as follows:

1. Choose the number of clusters, $\mathcal{C}$. Select a maximum change of membership between iterations, $\epsilon > 0$ and a maximum number of iterations, $I_m >> \mathcal{C}$, as ending conditions. Initialize an iteration counter to zero, $I = 0$. Finally, randomly initialize the centers of the clusters, $\mathbf{v}_i$.
2. Solve for the memberships of each vector:

$$\begin{array}{lll} u_{ik} & = & \dfrac{1}{\sum_{j=1}^{\mathcal{C}} \left[\dfrac{\|\mathbf{x}_k - \mathbf{v}_i\|}{\|\mathbf{x}_k - \mathbf{v}_j\|} \right]^2} \\ where & & k = 1, 2, \ldots, N \\ & & i = 1, 2, \ldots, \mathcal{C}. \end{array} \tag{16}$$

3. Solve for the centers for each cluster:

$$\begin{array}{lll} \mathbf{v}_i & = & \dfrac{\sum_{k=1}^{N} {u_{ik}}^2 \mathbf{x}_k}{\sum_{j=1}^{N} {u_{ij}}^2} \\ where & & i = 1, 2, \ldots, \mathcal{C}. \end{array} \tag{17}$$

4. Repeat Step 2.
5. Repeat Step 3.
6. If $\Delta u_{ik} > \epsilon$ & $I < I_M$, then increment I, and loop back to Step 4. Otherwise, stop.

Within our FCM clustering solution, we use this basic iteration with a slight variation so that the Picard iteration does not become as easily trapped in a poor local minimum caused by ramified data sets like the cantor set. We achieve this goal by eliminating a cluster center if the sum of the squares of its membership values, $U_i = \sum_{k=1}^{N} {u_{ik}}^2$, is less than a defined fraction, δ_U, of the average, U_i assigned to all the clusters. Next, to keep the number of clusters constant, the removed center is reinitialized with an arbitrary small distance from the center with the largest U_i. In effect, this modification attempts to prevent one cluster from containing a grossly disproportionate number of associated data points compared to the rest of the clusters. This modified FCM algorithm then proceeds as normal. However, there are some dangers using this method. Mainly, if one allows this elimination and reinitialization procedure to occur on every iteration, FCM does not have a chance to work before a cluster is divided. In an attempt to eliminate such cycles, we have empirically arrived at the following rules: the elimination/reinitialization procedure is executed only every 10th iteration and elimination/reinitializations are only allowed $\mathcal{C}$ times during the execution of the algorithm, giving each center an opportunity to be eliminated. Therefore, the following changes were made to the six steps listed above: initialization of a variable called $E = 0$ was added to Step 1, and Step 3 was replaced with

If $(I \; mod \; 10 = 0)$ & $(E \leq \mathcal{C})$, then

If $\min_i U_i < \delta_U \sum_{k=1}^{\mathcal{C}} U_k$, then $\mathbf{v}_i = \mathbf{v}_j + \delta$, where j achieves the maximum U_i and δ is a small but numerically significant randomly generated number to be added to each component of $\mathbf{v}_j$ to reinitialize the eliminated center near the largest cluster (effectively dividing the jth cluster once the membership values and centers are recalculated in the next iteration). Increment E and proceed to Step 2.

Continuing on with the solution for the new centers of each cluster

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} {u_{ik}}^2 \mathbf{x}_k}{\sum_{j=1}^{N} {u_{ij}}^2}, \tag{18}$$

where

$$i = 1, 2, \ldots, C. \tag{19}$$

Although all the data in this paper were processed using the above algorithm, a more efficient algorithm would detect limit cycles when the same centers are eliminated and reinitialized continuously. This suggests that continuing the process does not improve the clustering solution. Also, dividing the centers every 10th center appears to be excessive for small numbers of clusters. A rule of thumb: let the number of iterations between reinitializations be $\sqrt{\frac{N}{\mathcal{C}}}$, where N is the number of points in the data set and $\mathcal{C}$ is the number of centers. This allows more iterations between divisions for the algorithm to stabilize if N is large and increases the division rate when $\mathcal{C}$ is large.

APPENDIX B

FINDING THE DESIRED SUBOPTIMAL COVER: MINIMUM CLUSTER VOLUME PICARD ITERATION

For completeness, we summarize the MCV algorithm by Krishnapuram and Kim [18] here.

Due to the complexity of the MCV cost surface, numerous local minima exist. To overcome this limitation, we have

chosen to initialize the MCV center locations using our modified FCM algorithm, instead of the more traditional method of randomly selecting initial centers, see Appendix A. As with the FCM algorithm, the MCV algorithm is implemented through Picard-style iteration. Likewise, the centers are calculated using fuzzy membership values similar to the FCM algorithm. However, the membership values are determined through a more complex computation that is based on the Mahalanobis distance, Md_{ji}, which defines the distance between the center of a cluster, $\mathbf{v}_j$, and a data point of interest, $\mathbf{x}_i$, as:

$$Md_{ji} = (\mathbf{x}_i - \mathbf{v}_j)^T C_{fj}^{-1} (\mathbf{x}_i - \mathbf{v}_j). \tag{20}$$

The complexity of the update process used within the MCV iteration is due to its hybrid nature. Basically, the MCV algorithm is a hybrid of a fuzzy and hard clustering algorithm. The first case is a simple fuzzy membership update calculation, while the second case is a hard membership update calculation. The membership update calculation is chosen based on the value of the Mahalanobis distance, Md_{ji}. The fuzzy update process ($Md_{ji} > L \forall i$) is performed using the following calculation:

$$u_{sj} = \frac{\left(D_{sj}\right)^{1/(1-m)}}{\sum_{r=1}^{C} \left(D_{rj}\right)^{1/(1-m)}}, \tag{21}$$

where

$$D_{sj} = \frac{|C_{fs}|^{1/2}(Md_{sj} - L)}{\sum_{j=1}^{N} u_{sj}^m}. \tag{22}$$

For the hard clustering update ($i = (k| \min_{l=1,\ldots,C}(Md_{jl}) = Md_{jk})$ and $Md_{jk} < L$), the membership values are calculated as follows: $u_{kj} = 1$ for k such that Md_{kj} is the minimum of Md_{sj} for all centers, $s = 1, 2, \ldots, \mathcal{C}$; and all other membership values $u_{sj} = 0$ for $s \neq j$. Please note that adding the hard clustering procedure allows points close to the center to achieve a full membership within that cluster which increases their effect on the cluster's volume. This full membership and localization of each cluster's volume is the difference in the covers obtained using the FCM and MCV algorithms. With these two update procedures in hand, we proceed as in the FCM case with a Picard style iteration, alternating between calculating membership values and center locations until the maximum change in membership values is less than a defined value ϵ.

1. Select $\epsilon > 0$ (this is the ending condition). Finally, initialize the centers, v_i, to the results of FCM.
2. Calculate C_{fi}^{-1} and $|C_{fi}|$ for each cluster $i = 1, \ldots, \mathcal{C}$:

$$C_{fi} = \frac{\sum_{j=1}^{N} u_{ij}^m (\mathbf{x}_j - \mathbf{v}_i)(\mathbf{x}_j - \mathbf{v}_i)^T}{\sum_{j=1}^{N} u_{ij}^m}. \tag{23}$$

   Use LU decomposition or your own favorite algorithm for inverting and finding the determinant of C_{fi}.
3. Determine the new membership values, u_{ij}. For each data point, $\mathbf{x}_j$, find the Md_{ji} for all centers, $\mathbf{v}_i$ and determine if any $Md_{ji} < L$ for any center. If any $Md_{ji} < L$, find the index k such that Md_{jk} is the minimum of Md_{ji} over $i = 1, \ldots, \mathcal{C}$:

$$u_{ij} = \begin{cases} \frac{\left(D_{sj}\right)^{1/(1-m)}}{\sum_{r=1}^{\mathcal{C}} \left(D_{rj}\right)^{1/(1-m)}} & Md_{ji} \geq L, \forall i = 1, \ldots, \mathcal{C} \\ 1 & i = (k| \min_{l=1,\ldots,\mathcal{C}} \\ & (Md_{jl}) = Md_{jk}) \text{ and} \\ & Md_{jk} < L \\ 0 & i \neq k \text{ and } Md_{ji} < L \\ & \text{for any } i = 1, \ldots, \mathcal{C}. \end{cases} \tag{24}$$

4. Update the location of the centers, $\mathbf{v}_i$:

$$\mathbf{v}_i = \frac{\sum_{j=1}^{N} u_{ij}^m \mathbf{x}_j}{\sum_{j=1}^{N} u_{ij}^m}. \tag{25}$$

5. Repeat Steps 2 and 3 and find the change in membership values, Δu_{ik}.
6. If $\Delta u_{ik} > \epsilon$, then loop back to step 4. Otherwise, stop.

APPENDIX C
NOMENCLATURE

- $i, j, k, l,$ and s: used as counting indices throughout
- A: image or signal to be analyzed
- $D_h(A)$: Hausdorff-Besicovitch (HB) dimension definition
- $D_b(A)$: box dimension definition
- x_i: a component of a vector x
- $\mathbf{x}_i$: vector for the Ith data point
- $\mathbf{v}_k$: vector describing the center of the kth cluster
- u_{ik}: membership value for the kth point in the ith cluster
- N_i: number of hard clustered points within the ith cluster
- N: number of data points with the image of interest A
- h^S: s-dimensional Hausdroff measure
- L: dimension of the image of signal to be analyzed (A)
- $\hat{\mathcal{M}}^i$: scatter matrix for the ith cluster
- $\mathcal{Q}^i$: matrix of singular values
- σ_l^i: lth singular value of $\mathcal{Q}^i$
- W^i: directional unitary matrix on the left within the singular value decomposition
- w_l^i: lth column vector of W^i
- G^i: directional unitary matrix on the right within the singular value decomposition
- g_l^i: lth row vector of G^i
- $\mathcal{C}, \mathcal{C}_d$: number of covering elements for a given size, d, of covering element
- d: size of a covering element
- $d_e(x, y)$: Euclidean distance function between x and y
- $diam(C)$: diameter of the covering element C
- C_i: cover of the ith cluster
- $h^S(A)$: Hausdroff measure of A
- J_{fv}: cost function for the MCV problem
- J_{FCM}: cost function for the FCM problem
- m: fuzzifying exponent for MCV
- ϵ: maximum change in membership allowed for conversion of the clustering methods
- E: number of elimination/reinitializations that have occurred during FCM clustering
- I: iteration through the Picard iteration

- Md_{ji}: Mahalanobis distance
- D_{sj}: used in the calculation of the MCV Picard iteration—not related to $D_h(A)$
- C_{fi}: cover of the ith cluster in the MCV

ACKNOWLEDGMENTS

This work was supported by the Tank-Automotive Research, Development, Engineering and Center (TARDEC) part of the Tank-Automotive and Armanent Command (TACOM) within the US Department of the Army and by the US Department of Energy, Office of Science, Office of Basic Energy Sciences, under DOE Idaho Operations Office Contract DE-AC07-99ID13727.

REFERENCES

[1] C.R. Tolle and D. Gorsich, "Sub-Optimal Covers for Measuring Fractal Dimension," *Proc. Rocky Mountain NASA Space G. Consortium Conf.,* 1996.
[2] D. Gorsich, C.R. Tolle, R. Karlsen, and G. Gerhart, "Wavelet and Fractal Analysis of Ground Vehicle Signatures," *Proc. Seventh Ann. Ground Vehicle Survivability Symp.,* Mar. 1996.
[3] D. Gorsich, C.R. Tolle, R. Karlsen, and G. Gerhart, "Wavelet and Fractal Analysis of Ground Vehicle Images," *Proc. SPIE Symp.,* Aug. 1996.
[4] L.F. Jardine, "Fractal-Based Analysis and Synthesis of Multispectral Visual Texture for Camouflage," *Applications of Fractals and Choas,* pp. 101-116, New York: Springer-Verlag, 1993.
[5] D.E. Kreithen, S.D. Halversen, and G.J. Owirka, "Discriminating Targets from Clutter," *MIT Lincoln Laboratory J.,* vol. 6, no. 1, Spring, pp. 25-52, 1993.
[6] B.B. Mandelbrot, *The Fractal Geometry of Nature: Updated and Augmented.* New York: W.H. Freeman and Company, 1983.
[7] C.R. Tolle, T.R. McJunkin, D.T. Rohrbaugh, and R.A. LaViolette, "Optimal Cover-Based Definitions of Lacunarity for Ramified Data Sets," *Physica D,* submitted for publication Dec. 2000.
[8] B. Dubuc and S. Dubuc, "Error Bounds on the Estimation of Fractal Dimension," *Siam J. Numerical Analysis,* vol. 33, no. 2, pp. 602-626, Apr. 1996.
[9] B. Dubuc, "Evaluating the Fractal Dimension of Profiles," *Physics Rev. A,* vol. 39, no. 3, pp. 1500-1512, 1989.
[10] X.C. Jin, S.H. Ong, and Jayasooriah, "A Practical Method for Estimating Fractal Dimension," *Pattern Recognition Letters,* vol. 16, no. 5, pp. 457-464, May 1995.
[11] R.M. Summers, L.M. Pusanik, J.D. Malley, and J.M. Hoeg, "Fractal Analysis of Virtual Endoscopy Reconstructions," *Proc. SPIE Medical Imaging: Physiology and Function from Multidemensional Images,* C.-T. Chen and A.V. Clough, eds., vol. 3660, pp. 258-269, 1999.
[12] X. Jeng, L. Koehl, and C. Vasseur, "Design and Implementation of an Estimator of Fractal Dimension Using Fuzzy Techniques," *Pattern Recognition,* vol. 34, no. 1, pp. 151-169, Jan. 2001.
[13] K. Forotan-pour, P. Dutilleul, and D.L. Smith, "Advances in the Implementation of the Box-Counting Method of Fractal Dimension Estimation," *Applied Math. and Computation,* vol. 105, nos. 2-3, pp. 195-210, Nov. 1999.
[14] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. II, no. 7, pp. 674-693, July 1989.
[15] *Texturing and Modeling: A Procedural Approach.* D.S. Ebert, ed., pp. 256-260, Cambridge, Mass.: Academic Press Professional, 1994.
[16] H.-O. Peitgen, H. Jürgens, and D. Saupe, *Chaos and Fractals: New Frontiers of Science.* New York: Springer-Verlag, 1992.
[17] J.B. Bassingthwaighte, L.S. Liebovitch, and B.J. West, *Fractal Physiology.* New York: Am. Physiological Soc., 1994.
[18] R. Krishnapuram and J. Kim, "Clustering Algorithms on Volume Criteria," *IEEE Trans. Fuzzy Systems,* vol. 8, no. 2, pp. 228-236, Apr. 2000.
[19] R.L. Cannon, J.V. Dave, and J.C. Bezdek, "Efficient Implementation of Fuzzy C-Means Clustering Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, no. 2, pp. 248-255, 1986.
[20] P.V. O'Neil, *Advanced Engineering Mathematics,* pp. 57-61. Belmont, Calif.: Wadsworth Publishing, 1987.
[21] P. Hartman, *Ordinary Differential Equations.* New York: John Wiley & Sons, 1964.
[22] R.D. Blumofe, C.F. Joerg, B.C. Kuszmaul, C.E. Leiserson, K.H. Randall, and Y. Zhou, "Cilk: An Efficient Multithreaded Runtime System," *J. Parallel and Distributed Computing,* vol. 37, no. 1, pp. 55-69, Aug. 1996.
[23] M. Frigo, K.H. Randall, and C.E. Leiserson, "The Implementation of the Cilk-5 Multithreaded Language," *Proc. ACM SIGPLAN '98 Conf. Programming Language Design and Implementation (PLDI),* June 1998.

Charles R. Tolle received the BS degree in electrical engineering from the University of Utah in 1990, the MS degree from Arizona State University in 1994, and the PhD degree from Utah State University in 1998. He became an Arizona State University Industrial fellow for Honeywell Commercial Flight Systems Group (CFSG) in June of 1990 and a Rocky Mountain NASA Space Grant fellow in 1992. His research interests have included reactor off gas, biomedical control systems, aircraft guidance, target detection, human vision, fractals, signal motivation, and chaos theory. His research has been supported by General Electric, Hamilton Ventilators, Honeywell CFGS, Rocky Mountain NASA Space Grant Consortium, Tank Automotive Research Development Engineering Center (TARDEC) part of Tank-Automotive and Armaments Command (TACOM)-US Department of Army, Basic Energy Sciences-US Department of Energy, and Office of Industrial Technology-US Department of Energy. Dr. Tolle joined the Idaho National Engineering and Environmental Laboratory (INEEL) in 1997. His current research focuses on control systems for biological, welding, and high temperature processes, as well as complex/chaotic system modelling and signal motivation. Dr. Tolle is the president of the Rocky Mountain NASA Space Grant Consortium Fellows Association, as well as a trustee of the Rocky Mountain NASA Space Grant Consortium. He has authored/coauthored numerous papers in guidance, control, fractal analysis, and chaos theory. In addition, Dr. Tolle has two patents pending. He is a senior member of the IEEE, member of the IEEE Controls Society, and a member of the American Welding Society.

Timothy R. McJunkin received the BS and MS degrees in electrical engineering from Utah State University in 1992 and 1995. He was awarded a Rocky Mountain NASA Space Grant Fellowship in 1993. He was a design engineer for Compaq Computer Corporation Industry Standard Server Group from 1994 to 1999. Mr. McJunkin joined the Idaho National Engineering and Environmental Laboratory in 1999. His research interests are in nondestructive evaluation of materials, fuzzy logic, control systems, autonomous mobile vehicles, and (most recently) fractal texture characterization. He is a member of IEEE Control Systems Society, IEEE System, Man, and Cybernetics Society, IEEE Robotics and Automation Society, and IEEE Computer Society.

David J. Gorsich received the BSEE degree from Lawrence Technological University, Southfield, Michigan in 1990, the MS degree in applied mathematics, from Geroge Washington University in 1994, and the PhD mathematics from the Massachusetts Institute of Technology, Cambridge, in 2000. He is a senior research scientist, US Army Tank-Automotive and Armaments Command. Dr. Gorsich is currently a team leader for Vehicle Intelligence and Safety, director of TARDEC Robotics Laboratory, and director of the Automotive Research Center (ARC). His current research interests are approximation and spatial statistics, robotics, and vehicle intelligence.